In [1]:

```
from keras.models import Sequential
from keras.layers import LSTM, Input, Dense, GRU, Embedding, Dropout
from keras.optimizers import RMSprop
from keras import activations
from keras.callbacks import EarlyStopping, ModelCheckpoint, TensorBoard, ReduceLROnPlateau
#from keras.initializers import RandomUniform
#from keras.initializers import Initializer

import numpy as np
import operator
import seaborn as sns
import matplotlib.pyplot as plt
import tensorflow as tf
import math




sns.set_style("whitegrid")
current_palette = sns.color_palette('colorblind')
```

```
Using TensorFlow backend.
```

In [2]:

```
features = 20 #entspricht der Anzahl der Sensoren
timesteps = 22 # *0.05s --> definiert die Zeitspanne in der zeitliche Abhängigkeiten vom Netzwerk er
batchsize = 128
LSTM_size = 64 #ANzahl der LSTM-Zellen
Dense_size = 32
epochen = 100


name = 'NN2_3_1_2_My'

#init = RandomUniform(minval=-0.05, maxval=0.05)
```

In [3]:

```
# Um Vorhandenes, bereits trainiertes Model zu laden:

from keras.models import load_model
model = load_model('model/'+name)
```

Aufbau Model

In [4]:

```
model = Sequential()
model.add(LSTM(LSTM_size, dropout=0.3, recurrent_dropout=0.3 ,return_sequences=True,
               batch_input_shape=(None, timesteps, features)))
#model.add(LSTM(LSTM_size, dropout=0.3, recurrent_dropout=0.3 ,return_sequences=True))
model.add(LSTM(LSTM_size, dropout=0.3, recurrent_dropout=0.3 ,return_sequences=False))
```

```python
#model.add(Dense(Dense_size))
model.add(Dense(Dense_size))
model.add(Dense(1, activation='linear'))

model.compile(loss='mse', optimizer='rmsprop')
```

Trainingsdaten laden

In [5]:

```python
x_train = np.load('Regression_Daten/x_train.npy').astype('float32')
x_val = np.load('Regression_Daten/x_val.npy').astype('float32')
x_test = np.load('Regression_Daten/x_test.npy').astype('float32')

y_train = np.load('Regression_Daten/y_My_train2.npy').astype('float32')
y_val = np.load('Regression_Daten/y_My_val2.npy').astype('float32')
y_test = np.load('Regression_Daten/y_My_test2.npy').astype('float32')
```

Model trainieren

In [6]:

```python
model.fit(x_train, y_train,
          batch_size=batchsize , epochs=epochen, validation_data=(x_val, y_val))
```

```
Train on 11392 samples, validate on 2944 samples
Epoch 1/100
11392/11392 [==============================] - 7s 603us/step - loss: 279.3591 - val_loss:
98.7150
Epoch 2/100
11392/11392 [==============================] - 5s 421us/step - loss: 122.8152 - val_loss:
77.8337
Epoch 3/100
11392/11392 [==============================] - 5s 426us/step - loss: 96.6122 - val_loss:
64.6356
Epoch 4/100
11392/11392 [==============================] - 5s 414us/step - loss: 79.9087 - val_loss:
50.2676
Epoch 5/100
11392/11392 [==============================] - 5s 410us/step - loss: 71.5979 - val_loss:
55.1677
Epoch 6/100
11392/11392 [==============================] - 5s 410us/step - loss: 65.9927 - val_loss:
55.8478
Epoch 7/100
11392/11392 [==============================] - 5s 401us/step - loss: 61.2959 - val_loss:
58.0034
Epoch 8/100
11392/11392 [==============================] - 5s 400us/step - loss: 58.6511 - val_loss:
55.3179
Epoch 9/100
11392/11392 [==============================] - 5s 399us/step - loss: 55.0659 - val_loss:
62.2347
Epoch 10/100
11392/11392 [==============================] - 5s 408us/step - loss: 54.1221 - val_loss:
```

```
57.2316
Epoch 11/100
11392/11392 [==============================] - 5s 405us/step - loss: 51.3858 - val_loss:
55.6261
Epoch 12/100
11392/11392 [==============================] - 5s 406us/step - loss: 49.8369 - val_loss:
54.2957
Epoch 13/100
11392/11392 [==============================] - 5s 405us/step - loss: 47.7865 - val_loss:
66.3647
Epoch 14/100
11392/11392 [==============================] - 5s 415us/step - loss: 46.9451 - val_loss:
63.7328
Epoch 15/100
11392/11392 [==============================] - 5s 433us/step - loss: 44.5463 - val_loss:
60.1103
Epoch 16/100
11392/11392 [==============================] - 5s 430us/step - loss: 43.3841 - val_loss:
58.5211
Epoch 17/100
11392/11392 [==============================] - 5s 423us/step - loss: 42.7144 - val_loss:
57.9150
Epoch 18/100
11392/11392 [==============================] - 5s 425us/step - loss: 41.8428 - val_loss:
59.6443
Epoch 19/100
11392/11392 [==============================] - 5s 413us/step - loss: 40.9100 - val_loss:
54.0069
Epoch 20/100
11392/11392 [==============================] - 5s 407us/step - loss: 39.2634 - val_loss:
60.6268
Epoch 21/100
11392/11392 [==============================] - 5s 411us/step - loss: 38.2218 - val_loss:
57.5654
Epoch 22/100
11392/11392 [==============================] - 5s 413us/step - loss: 38.5506 - val_loss:
66.4379
Epoch 23/100
11392/11392 [==============================] - 5s 417us/step - loss: 37.5472 - val_loss:
62.1062
Epoch 24/100
11392/11392 [==============================] - 5s 424us/step - loss: 37.4967 - val_loss:
54.7814
Epoch 25/100
11392/11392 [==============================] - 5s 431us/step - loss: 36.9841 - val_loss:
50.4259
Epoch 26/100
11392/11392 [==============================] - 5s 431us/step - loss: 37.3164 - val_loss:
61.9157
Epoch 27/100
11392/11392 [==============================] - 5s 416us/step - loss: 35.5292 - val_loss:
50.5293
```

```
Epoch 28/100
11392/11392 [==============================] - 5s 435us/step - loss: 35.0322 - val_loss:
55.0749
Epoch 29/100
11392/11392 [==============================] - 5s 456us/step - loss: 34.3357 - val_loss:
54.2442
Epoch 30/100
11392/11392 [==============================] - 5s 416us/step - loss: 34.3149 - val_loss:
64.4429
Epoch 31/100
11392/11392 [==============================] - 5s 422us/step - loss: 33.7687 - val_loss:
50.5976
Epoch 32/100
11392/11392 [==============================] - 5s 414us/step - loss: 32.9956 - val_loss:
55.9269
Epoch 33/100
11392/11392 [==============================] - 5s 409us/step - loss: 33.3149 - val_loss:
55.0050
Epoch 34/100
11392/11392 [==============================] - 5s 405us/step - loss: 32.2261 - val_loss:
54.4166
Epoch 35/100
11392/11392 [==============================] - 4s 319us/step - loss: 32.3605 - val_loss:
61.3646
Epoch 36/100
11392/11392 [==============================] - 3s 294us/step - loss: 32.2855 - val_loss:
57.8536
Epoch 37/100
11392/11392 [==============================] - 3s 302us/step - loss: 32.4913 - val_loss:
62.9873
Epoch 38/100
11392/11392 [==============================] - 3s 300us/step - loss: 31.6214 - val_loss:
52.7488
Epoch 39/100
11392/11392 [==============================] - 4s 308us/step - loss: 31.1337 - val_loss:
52.2470
Epoch 40/100
11392/11392 [==============================] - 4s 308us/step - loss: 32.0689 - val_loss:
47.1553
Epoch 41/100
11392/11392 [==============================] - 3s 289us/step - loss: 30.5829 - val_loss:
60.5837
Epoch 42/100
11392/11392 [==============================] - 3s 296us/step - loss: 30.5551 - val_loss:
58.6780
Epoch 43/100
11392/11392 [==============================] - 4s 308us/step - loss: 30.6159 - val_loss:
61.9722
Epoch 44/100
11392/11392 [==============================] - 4s 318us/step - loss: 30.3976 - val_loss:
65.0040
Epoch 45/100
```

```
11392/11392 [==============================] - 4s 316us/step - loss: 30.2660 - val_loss:
56.1989
Epoch 46/100
11392/11392 [==============================] - 4s 312us/step - loss: 31.0427 - val_loss:
50.6001
Epoch 47/100
11392/11392 [==============================] - 4s 313us/step - loss: 30.1001 - val_loss:
56.8642
Epoch 48/100
11392/11392 [==============================] - 4s 310us/step - loss: 29.9186 - val_loss:
47.8310
Epoch 49/100
11392/11392 [==============================] - 3s 303us/step - loss: 29.2742 - val_loss:
55.2775
Epoch 50/100
11392/11392 [==============================] - 4s 331us/step - loss: 28.9004 - val_loss:
61.0157
Epoch 51/100
11392/11392 [==============================] - 4s 325us/step - loss: 29.6829 - val_loss:
61.4984
Epoch 52/100
11392/11392 [==============================] - 4s 318us/step - loss: 29.1902 - val_loss:
61.0511
Epoch 53/100
11392/11392 [==============================] - 4s 311us/step - loss: 29.1574 - val_loss:
56.8756
Epoch 54/100
11392/11392 [==============================] - 3s 301us/step - loss: 28.9189 - val_loss:
54.3627
Epoch 55/100
11392/11392 [==============================] - 3s 290us/step - loss: 28.9073 - val_loss:
56.5071
Epoch 56/100
11392/11392 [==============================] - 3s 291us/step - loss: 28.9836 - val_loss:
55.9576
Epoch 57/100
11392/11392 [==============================] - 4s 312us/step - loss: 28.1583 - val_loss:
54.6521
Epoch 58/100
11392/11392 [==============================] - 3s 292us/step - loss: 28.5524 - val_loss:
52.5914
Epoch 59/100
11392/11392 [==============================] - 3s 299us/step - loss: 27.8316 - val_loss:
53.9917
Epoch 60/100
11392/11392 [==============================] - 4s 311us/step - loss: 28.2773 - val_loss:
54.2857
Epoch 61/100
11392/11392 [==============================] - 3s 295us/step - loss: 27.9633 - val_loss:
52.8168
Epoch 62/100
11392/11392 [==============================] - 3s 290us/step - loss: 27.3058 - val_loss:
```

```
61.0164
Epoch 63/100
11392/11392 [==============================] - 3s 305us/step - loss: 27.8978 - val_loss:
58.7197
Epoch 64/100
11392/11392 [==============================] - 4s 308us/step - loss: 27.4142 - val_loss:
60.9932
Epoch 65/100
11392/11392 [==============================] - 3s 293us/step - loss: 27.2432 - val_loss:
54.4836
Epoch 66/100
11392/11392 [==============================] - 3s 276us/step - loss: 27.1231 - val_loss:
60.3286
Epoch 67/100
11392/11392 [==============================] - 3s 294us/step - loss: 26.7569 - val_loss:
57.1975
Epoch 68/100
11392/11392 [==============================] - 4s 309us/step - loss: 27.2604 - val_loss:
54.9766
Epoch 69/100
11392/11392 [==============================] - 4s 310us/step - loss: 26.7907 - val_loss:
54.0539
Epoch 70/100
11392/11392 [==============================] - 4s 317us/step - loss: 27.2432 - val_loss:
56.7998
Epoch 71/100
11392/11392 [==============================] - 3s 302us/step - loss: 26.4502 - val_loss:
58.4899
Epoch 72/100
11392/11392 [==============================] - 4s 315us/step - loss: 26.2015 - val_loss:
65.6252
Epoch 73/100
11392/11392 [==============================] - 4s 313us/step - loss: 26.6838 - val_loss:
60.2511
Epoch 74/100
11392/11392 [==============================] - 4s 313us/step - loss: 26.2505 - val_loss:
64.0698
Epoch 75/100
11392/11392 [==============================] - 4s 317us/step - loss: 26.3655 - val_loss:
62.6974
Epoch 76/100
11392/11392 [==============================] - 4s 321us/step - loss: 26.2411 - val_loss:
56.4077
Epoch 77/100
11392/11392 [==============================] - 4s 316us/step - loss: 26.3152 - val_loss:
59.3771
Epoch 78/100
11392/11392 [==============================] - 4s 324us/step - loss: 25.6011 - val_loss:
55.0945
Epoch 79/100
11392/11392 [==============================] - 3s 306us/step - loss: 26.5006 - val_loss:
55.7898
```

```
Epoch 80/100
11392/11392 [==============================] - 4s 311us/step - loss: 25.4721 - val_loss:
56.0639
Epoch 81/100
11392/11392 [==============================] - 4s 309us/step - loss: 25.9024 - val_loss:
57.8958
Epoch 82/100
11392/11392 [==============================] - 4s 308us/step - loss: 25.4862 - val_loss:
53.1854
Epoch 83/100
11392/11392 [==============================] - 3s 301us/step - loss: 25.1659 - val_loss:
61.1488
Epoch 84/100
11392/11392 [==============================] - 3s 300us/step - loss: 25.3763 - val_loss:
58.2732
Epoch 85/100
11392/11392 [==============================] - 4s 314us/step - loss: 25.1710 - val_loss:
53.8375
Epoch 86/100
11392/11392 [==============================] - 3s 306us/step - loss: 25.4414 - val_loss:
55.3229
Epoch 87/100
11392/11392 [==============================] - 4s 308us/step - loss: 24.8794 - val_loss:
61.5275
Epoch 88/100
11392/11392 [==============================] - 3s 298us/step - loss: 24.8854 - val_loss:
58.8597
Epoch 89/100
11392/11392 [==============================] - 3s 306us/step - loss: 24.7416 - val_loss:
55.5428
Epoch 90/100
11392/11392 [==============================] - 3s 302us/step - loss: 24.9442 - val_loss:
64.5886
Epoch 91/100
11392/11392 [==============================] - 3s 295us/step - loss: 24.0022 - val_loss:
59.9687
Epoch 92/100
11392/11392 [==============================] - 3s 303us/step - loss: 25.4211 - val_loss:
57.1206
Epoch 93/100
11392/11392 [==============================] - 3s 303us/step - loss: 24.9856 - val_loss:
53.1244
Epoch 94/100
11392/11392 [==============================] - 3s 297us/step - loss: 24.2168 - val_loss:
58.6777
Epoch 95/100
11392/11392 [==============================] - 4s 309us/step - loss: 24.1302 - val_loss:
45.6429
Epoch 96/100
11392/11392 [==============================] - 4s 311us/step - loss: 24.0291 - val_loss:
56.2345
Epoch 97/100
```

```
11392/11392 [==============================] - 3s 307us/step - loss: 24.2488 - val_loss:
52.7266
Epoch 98/100
11392/11392 [==============================] - 4s 309us/step - loss: 24.5457 - val_loss:
58.5404
Epoch 99/100
11392/11392 [==============================] - 3s 304us/step - loss: 24.9657 - val_loss:
56.6421
Epoch 100/100
11392/11392 [==============================] - 3s 307us/step - loss: 23.6816 - val_loss:
55.8901
```

Out [6]:

```
<keras.callbacks.History at 0x1a2d1d34a8>
```

In [7]:

```
history_dict = model.history.history
history_dict
```

Out [7]:

```
{'val_loss': [98.71502005535623,
  77.83374396614407,
  64.63562484409498,
  50.2675953740659,
  55.167741651120394,
  55.847750622293226,
  58.00338247547979,
  55.317910277325176,
  62.23474884033203,
  57.23156215833581,
  55.626137194426164,
  54.29565255538277,
  66.36467481696087,
  63.732793725055195,
  60.110324279121734,
  58.52105103368344,
  57.91496297587519,
  59.64426264555558,
  54.006877982098125,
  60.62676939756974,
  57.56540961887526,
  66.43794262927511,
  62.10623392851456,
  54.78142414922299,
  50.42590983017631,
  61.915669565615445,
  50.529311843540356,
  55.07486824367357,
  54.244211984717325,
  64.44291342859682,
  50.59762593974238,
```

55.92691031746242,
55.004969016365386,
54.41656983417013,
61.36456191021463,
57.853572306425676,
62.987295150756836,
52.74875703065292,
52.24700600167979,
47.15527455703072,
60.58366505996041,
58.67797515703284,
61.97217733963676,
65.00403031058933,
56.19886365144149,
50.6000803242559,
56.86415154000987,
47.831022345501445,
55.27748373280401,
61.015735335971996,
61.498399402784266,
61.051063288813054,
56.87564493262249,
54.3627345458321,
56.5071207544078,
55.9576128254766,
54.65211005832838,
52.59144890826681,
53.99174084870712,
54.285660536392875,
52.81684514750605,
61.0163864467455,
58.719731952833094,
60.99321929268215,
54.48359912374745,
60.32862700586734,
57.19753476847773,
54.97663365239682,
54.05394794629968,
56.79981401692266,
58.4898566370425,
65.62516714178997,
60.251069276229195,
64.06977993509044,
62.697374136551566,
56.40769527269446,
59.37710927880329,
55.09447773643162,
55.78984521782917,
56.06386752750563,
57.895833513011105,
53.18544420988663,
61.14875909556513,

```
 58.27323639911154,
 53.83747075951618,
 55.322909313699476,
 61.527463664179265,
 58.85969841998556,
 55.542844150377356,
 64.58859306833018,
 59.9686801744544,
 57.12057163404382,
 53.124360789423406,
 58.67772516996964,
 45.64288956186046,
 56.23453737341839,
 52.72660524948783,
 58.54035137010657,
 56.642098965852156,
 55.89014476278554],
'loss': [279.35907436756605,
 122.81522952304798,
 96.61222556467807,
 79.9086873772439,
 71.59792036420843,
 65.99265044994569,
 61.295861619242125,
 58.65109728695302,
 55.065917240099964,
 54.12205616811688,
 51.38579475745726,
 49.83688200189826,
 47.78652628352133,
 46.94510354888573,
 44.54632958401455,
 43.38406875696075,
 42.71442634068178,
 41.84279242526279,
 40.91002760040626,
 39.26338692997279,
 38.221848477138565,
 38.550570713000354,
 37.54719136270244,
 37.49674036261741,
 36.98413908883427,
 37.31644349687555,
 35.52923082501701,
 35.03218590811397,
 34.33569003758806,
 34.314900280384535,
 33.7686931310075,
 32.99561594845204,
 33.314878013696564,
 32.22606146737431,
 32.360509465249734,
```

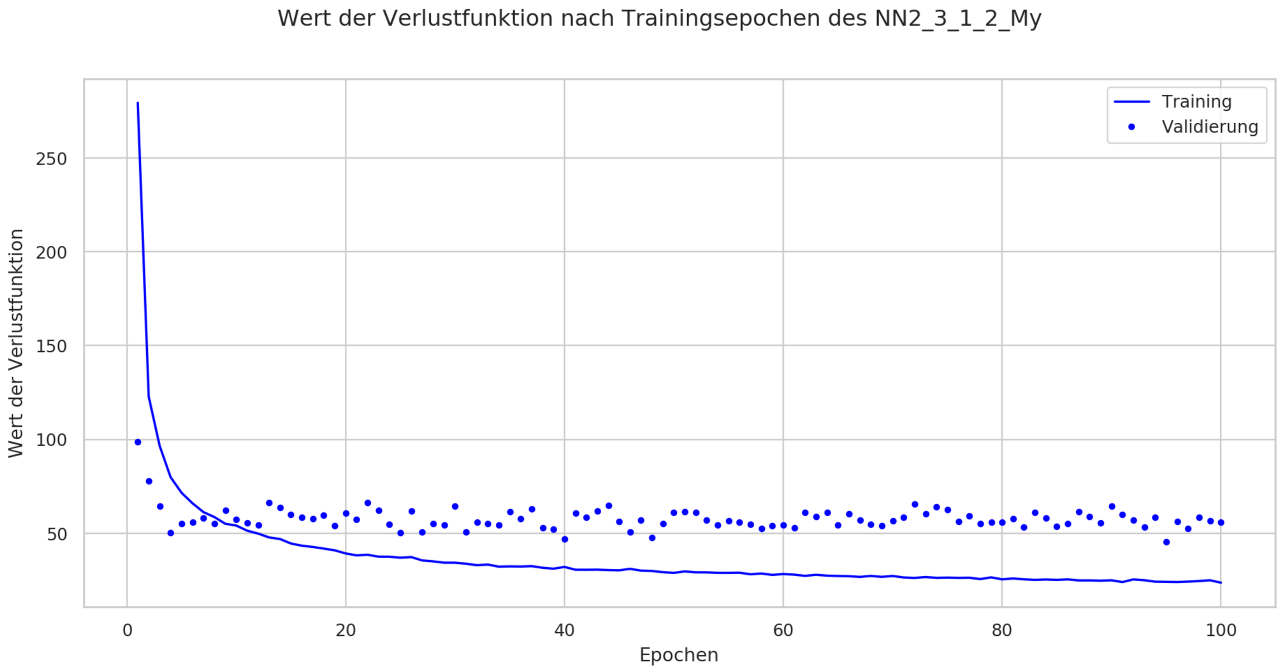
32.28547081250823,
32.49129368214125,
31.62138186679797,
31.13365949137827,
32.068920371237766,
30.58292001017024,
30.55513101213434,
30.615861335497225,
30.39756477013063,
30.265999590412953,
31.042690084221658,
30.100133531549005,
29.91859221726321,
29.274191288465865,
28.9004457452324,
29.682948616113556,
29.190192833375395,
29.157423619473917,
28.91888202710098,
28.907344989562304,
28.983633212828906,
28.15829397051522,
28.552371550142094,
27.83156412103203,
28.277343096358052,
27.963330343867955,
27.3058472322614,
27.897753683368812,
27.41424365525835,
27.243248971660485,
27.1231058956532,
26.756857121928356,
27.26043718316582,
26.790743645657315,
27.243246935726553,
26.450165469994705,
26.201546272535,
26.683785642130992,
26.250479901774547,
26.365537707725267,
26.24112122782161,
26.31516780210345,
25.601135703954803,
26.50055304537998,
25.4721244211947,
25.90237214592066,
25.48615489916855,
25.16593240887931,
25.37632001383921,
25.17098692561803,
25.44138357612524,
24.879355634196422,

```
24.88543866189678,
24.741644419980855,
24.944241898783137,
24.00215196073725,
25.421069905999,
24.985630378294527,
24.216765318024024,
24.130192906669016,
24.029090645607937,
24.248774871397554,
24.54571087440748,
24.965701692559747,
23.681568938694642]}
```

Analysiere Trainingsergebnisse

In [8]:

```python
sns.set_context("paper")

loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']
epochs = range(1, len(loss_values)+1)

plt.figure(num=None, figsize=(11,5), dpi=200, facecolor='w', edgecolor='k')
#f, (ax1,ax2) = plt.subplots(1,2, figsize=(11, 5), dpi=100, facecolor='w', edgecolor='k')
#f.suptitle('Trainingsverlauf '+name)
plt.plot(epochs, loss_values, 'b',label='Training')
plt.plot(epochs, val_loss_values, 'b.',label='Validierung')
plt.suptitle('Wert der Verlustfunktion nach Trainingsepochen des '+name)
plt.xlabel('Epochen')
plt.ylabel('Wert der Verlustfunktion')

plt.legend()#bbox_to_anchor=(0.9, 0., 0.5, 0.5), borderaxespad=1)
plt.show()
```



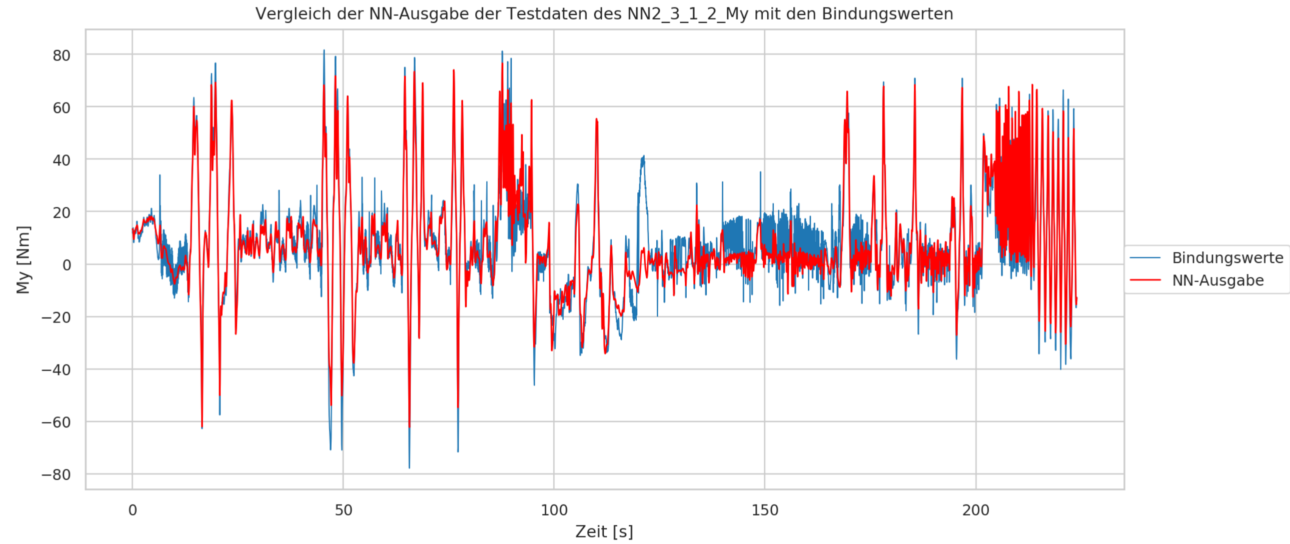
Anwendung des trainierten Models auf 'unbekannte' Trainingsdaten

In [6]:

```
predictions = model.predict(x_test,batch_size=batchsize)
y_real = y_test
```

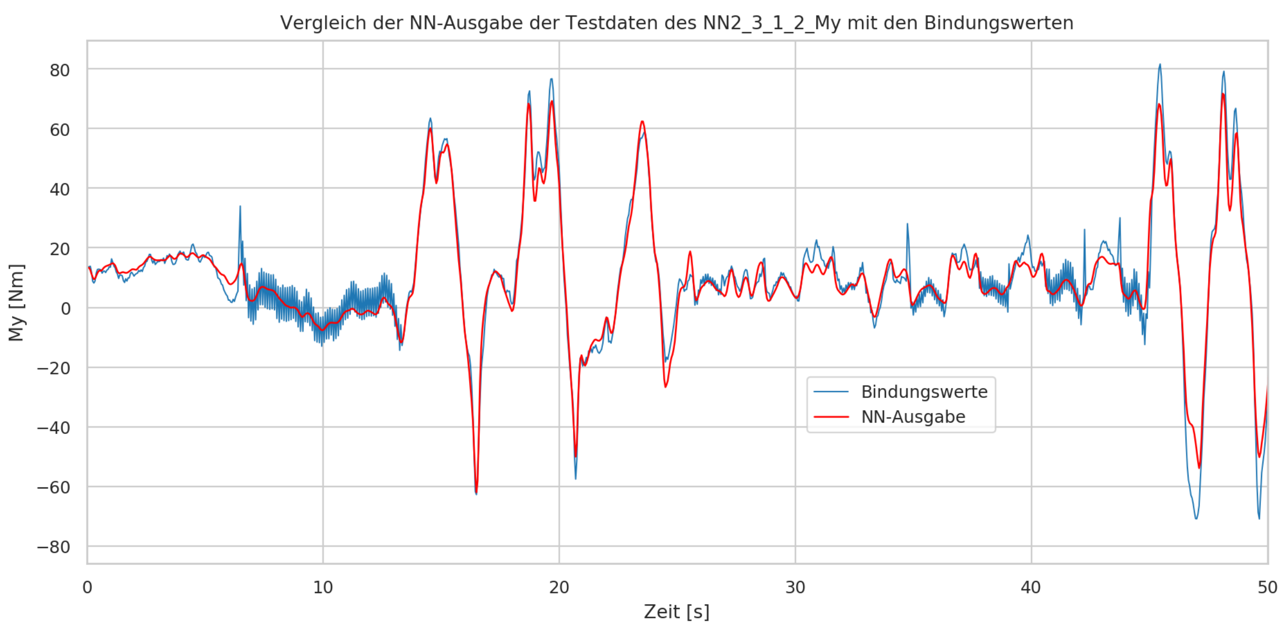
In [27]:

```
plt.figure(num=None, figsize=(11,5), dpi=200, facecolor='w', edgecolor='k')
plt.plot(np.linspace(0,len(predictions)*0.05-0.05,len(predictions)), y_real, label='Bindungswerte',
plt.plot(np.linspace(0,len(predictions)*0.05-0.05,len(predictions)), predictions,'r', label='NN-Ausg

plt.title('Vergleich der NN-Ausgabe der Testdaten des ' +name +' mit den Bindungswerten')
plt.xlabel('Zeit [s]')
plt.ylabel('My [Nm]')
plt.legend(bbox_to_anchor=(0.61, 0.15, 0.55, 0.38), borderaxespad=0)
#plt.xlim(left=140, right=220)
#plt.xlim(left=150, right=200)
#plt.ylim(bottom=-10, top=10)
plt.show()
```



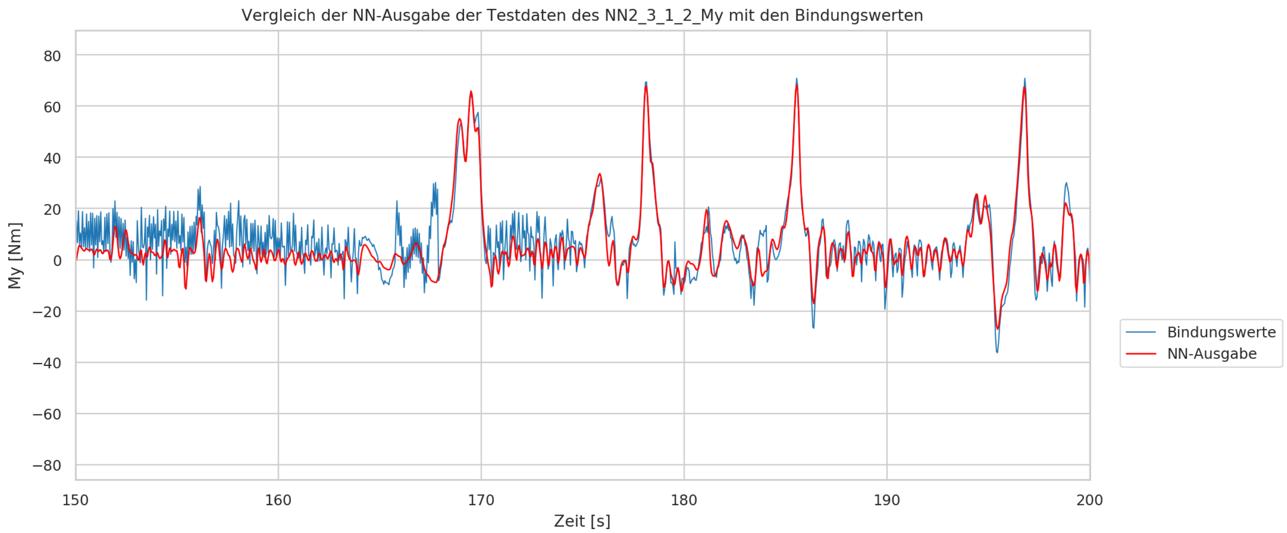
In [28]:

```
plt.figure(num=None, figsize=(11,5), dpi=200, facecolor='w', edgecolor='k')
plt.plot(np.linspace(0,len(predictions)*0.05-0.05,len(predictions)), y_real, label='Bindungswerte',
plt.plot(np.linspace(0,len(predictions)*0.05-0.05,len(predictions)), predictions,'r', label='NN-Ausg

plt.title('Vergleich der NN-Ausgabe der Testdaten des ' +name +' mit den Bindungswerten')
plt.xlabel('Zeit [s]')
plt.ylabel('My [Nm]')
plt.legend(bbox_to_anchor=(0.61, 0.25, 0.58, 0.45), borderaxespad=0)
plt.xlim(left=0, right=50)
#plt.xlim(left=150, right=200)
#plt.ylim(bottom=-10, top=10)
plt.show()
```



In [29]:

```
plt.figure(num=None, figsize=(11,5), dpi=200, facecolor='w', edgecolor='k')
plt.plot(np.linspace(0,len(predictions)*0.05-0.05,len(predictions)), y_real, label='Bindungswerte',
plt.plot(np.linspace(0,len(predictions)*0.05-0.05,len(predictions)), predictions,'r', label='NN-Ausg

plt.title('Vergleich der NN-Ausgabe der Testdaten des ' +name +' mit den Bindungswerten')
plt.xlabel('Zeit [s]')
plt.ylabel('My [Nm]')
plt.legend(bbox_to_anchor=(0.61, 0.25, 0.58, 0.45), borderaxespad=0)
#plt.xlim(left=0, right=50)
plt.xlim(left=150, right=200)
#plt.ylim(bottom=-10, top=10)
plt.show()
```



Zusammenfassung

In [30]:

```
# Nach welcher Epoche sollte das Training optimalerweise abgeschlossen werden?
# Vorgehen: einmal Netzwerk berechnen in 25 Epochen -> optimale Epochenanzahl anhand 'min_index_val_
#           danach Netzwerk neu berechnen lassen.
min_index_val_loss, min_value_val_loss = min(enumerate(history_dict['val_loss']), key=operator.itemg
#max_index_val_acc, max_value_val_acc = max(enumerate(history_dict['val_acc']), key=operator.itemget
print('Ergebnisse der Validierungsdaten:')
print('  opimale Epochenanzahl:                  '+str(min_index_val_loss+1))
print('  minimaler Verlust:                      '+str(min_value_val_loss)+'\n')

print('Ergebnisse der Trainingdaten zur optimalen Epochenzahl:')
print('  Verlust:                                '+str(history_dict['loss'][min_index_val_loss]) + '\
```

```
Ergebnisse der Validierungsdaten:
  opimale Epochenanzahl:                 95
```

```
  minimaler Verlust:                     45.64288956186046

Ergebnisse der Trainingdaten zur optimalen Epochenzahl:
  Verlust:                               24.130192906669016
```

In [31]:

```python
# Beurteilung der Testdaten: Vergleich von 'predictions' mit y_real
#                   - Kreuzkorrelation
#                   - Euklidsche Distanz
#   Kreuzkorrelation
print('Vergleich der Vorhersagewerte mit den Bindungswerten:')
print('  Korrelationskoeffizient:              '+ str(np.corrcoef(np.transpose(predictions),np.tran

#   Euklidsche Distanz
summe=0
for i in range(len(predictions)):
    summe+=math.pow(predictions[i]-y_real[i],2)
print('  Euklidsche Distanz:                   '+ str(math.sqrt(summe))+'\n\n')

print(model.summary())
```

```
Vergleich der Vorhersagewerte mit den Bindungswerten:
  Korrelationskoeffizient:               0.9346686121322699
  Euklidsche Distanz:                    481.1862537023323


_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 22, 64)            21760
_________________________________________________________________
lstm_2 (LSTM)                (None, 64)                33024
_________________________________________________________________
dense_1 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 33
=================================================================
Total params: 56,897
Trainable params: 56,897
Non-trainable params: 0
_________________________________________________________________
None
```

In [16]:

```python
print(name)
```

```
NN2_3_1_2_My
```